NAMA : MAUGY AL KAUTSAR

NIM : G.211.21.0086

PRODI : Teknik Informatika

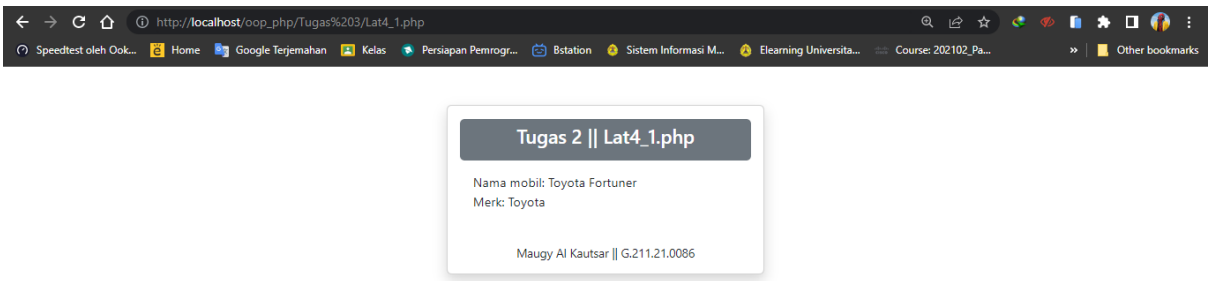
TUGAS PEMOGRAMAN FRAMEWORK OOP MVC

Latihan 1

```php
<?php
//class mobil
class Mobil
{
    public $nama;
    public $merk;
    function getInfo($a)
    {
        echo "Nama mobil: " . $a->nama . "<br/>";
        echo "Merk: " . $a->merk . "<br/>";
    }
}

?>
<div class="container mt-5 bg-gray-300">
    <div class="card pt-3 px-3 shadow mx-auto" style="width: 400px">
        <div class="card-header text-center text-bg-secondary border-0 rounded">
            <h4>Tugas 2 || Lat4_1.php </h4>
        </div>
        <div class="card-body">
            <p>
                <?php
                $a = new Mobil();
                $a->nama = "Toyota Fortuner";
                $a->merk = "Toyota";
                $a->getInfo($a);
                ?>
            </p>
        </div>
```

Jawab :

a.



b. Error, karena di php tidak bisa mengoverload method

c. Kesimpulan :

1. Cara membuat class pada php
   <?php
   Class nama_class{
   }
   ?>
2. Cara menuliskan property
   Modifier $nama_class;
3. Penulisan method
   Modifier function nama_method(){
   Isi method;
   }
4. Cara inisiasi object
   $nama_object = new nama_class();
5. Cara mengisi property atau mendefinisikan property
   $nama_object->properties='aaa';
6. Cara memanggil/menjalankan method pada suatu class
   $nama_object->nama_methode();

Latihan 2

A.Menampilkan construct pada Lat4_2a

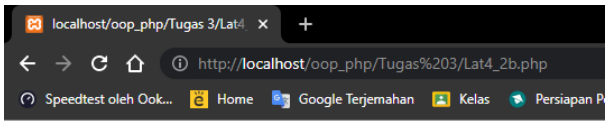
2a

```php
<?php
class mahasiswa
{
    public $nama;
    public $nim;
    public $prodi;
    public $semester;
    function __construct($a, $b, $c, $d)
    {
        $this->nama = $a;
        $this->nim = $b;
        $this->prodi = $c;
        $this->semester = $d;

        echo "Kelas telah dibuat<br/><br/>";
    }
    function cetak()
    {
        echo $this->nama . "<br/>" . $this->nim . "<br/>";
        echo $this->prodi . "<br/>" . $this->semester . "<br/></br>";
    }
    function __destruct()
    {
        echo "Kelas telah dihancurkan";
    }
}
class mahasiswi
{
    public $nama;
    public $nim;
    public $prodi;
    public $semester;
    function __construct($a, $b, $c, $d)
    {
        $this->nama = $a;
        $this->nim = $b;
        $this->prodi = $c;
        $this->semester = $d;

        echo "Kelas telah dibuat<br/><br/>";
    }
    function cetak()
    {
        echo $this->nama . "<br/>" . $this->nim . "<br/>";
        echo $this->prodi . "<br/>" . $this->semester . "<br/></br>";
    }
    function __destruct()
    {
        echo "Kelas telah dihancurkan";
    }
}
```

2b

```php
<?php
require_once("lat4_2a.php");
$mhs2 = new mahasiswa("Maugy Al Kautsar", "G.211.21.0086", "FTIK", "Semester 3");
$mhs2->cetak();
```

B.Tampilan



Kelas telah dibuat

Maugy Al Kautsar
G.211.21.0086
FTIK
Semester 3

Kelas telah dihancurkan

C.Kesimpulan :

Constructor dalam OOP PHP tidak dapat di Override

Latihan 3

A. Program tersbut mengalami beberapa error, Ha itu terjadi dikarenakan property yang bermodifilier private hanya bisa digunakan pada class mahasiswa sendiri

B. Modifier protected dan Public

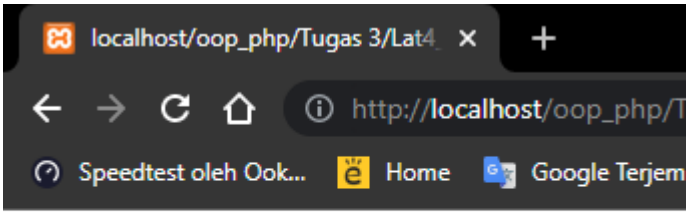
b.1 Protected

b.2 Public

Tampilan :

```php
<?php
class mahasiswa
{
    public $nama;
    public $nim;
    function __construct()
    {
    }
    function setNama($a)
    {
        $this->nama = $a;
    }
    function setNim($b)
    {
        $this->nim = $b;
    }
    function getNama()
    {
        return $this->nama;
    }
    function getNim()
    {
        return $this->nim;
    }
    function destruct()
    {
    }
}
```

```php
<?php
require_once("lat4_3a.php");
$mhs1 = new mahasiswa();
$mhs1->nama = "Maugy Al Kautsar </br></br>";
$mhs1->nim = "G.211.21.0086";
echo $mhs1->nama;
echo $mhs1->nim;
```



Maugy Al Kautsar

G.211.21.0086

C. Kesimpulan :

I. Penggunaan Modifier

Modifier keterangan public

Untuk mendefinisikan data atau metode yang akan terlihat dari luar oleh siapapun dan dimanapun.

A. Private

Untuk mendefinisikan data atau metode yang akan terlihat pada class/object itu sendiri.

B. Protected

Untuk mendefinisikan data atau metode untuk tidak terlihat dari luar (seperti private), tetapi akan dapat diakses oleh “anak” dari class tersebut.

II. Modifier Protected dan Private, Propertis bisa dipanggil dengan mengimplementasikan setter-getter.

Latihan 4

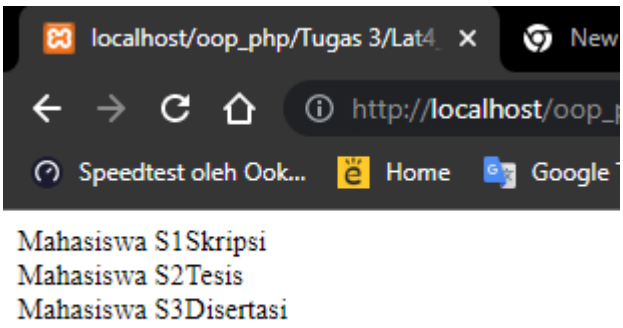
A. Tampilan

B. Kesimpulan : Pada PHP OOP, Class asisten (child) bisa memanggil methode dari mahasiswa (parent)

Latihan 5

```php
<?php
abstract class mahasiswa
{
    abstract protected function getTugasAkhir();
    abstract protected function getProgram($postfix);
    public function tugasAkhir()
    {
        print $this->getTugasAkhir() . "<br>";
    }
}
class sarjana extends mahasiswa
{
    protected function getTugasAkhir()
    {
        return "Skripsi";
    }
    public function getProgram($postfix)
    {
        print "{$postfix} S1";
    }
}
class magister extends mahasiswa
{
    public function getTugasAkhir()
    {
        return "Tesis";
    }
    public function getProgram($postfix)
    {
        print "{$postfix} S2";
    }
}
class Doctor extends mahasiswa
{
    public function getTugasAkhir()
    {
        return "Disertasi";
    }
    public function getProgram($postfix)
    {
        print "{$postfix} S3";
    }
}
```

```php
<?php
require_once("lat4_5a.php");
$s = new sarjana;
$s->getProgram('Mahasiswa') . "<br>";
$s->tugasAkhir();
$m = new magister;
$m->getProgram('Mahasiswa') . "<br>";
$m->tugasAkhir();
$d = new Doctor;
$d->getProgram('Mahasiswa') . "<br>";
$d->tugasAkhir();
```



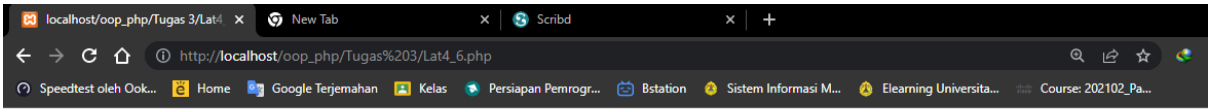
Kesimpulan : Class anak yang mewarisi super class harus menuliskan semua method abstrak dari super classnya

Latihan 6

A. Maksud dari program diatas adalah penggunaan object interface

B.

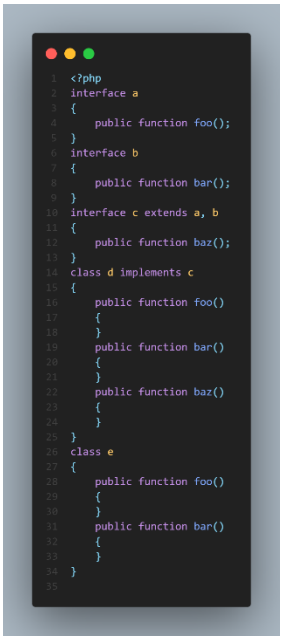
```php
<?php
interface a
{
    public function foo();
}
interface b
{
    public function bar();
}
interface c extends a, b
{
    public function baz();
}
class d implements c
{
    public function foo()
    {
    }
    public function bar()
    {
    }
}
```



Fatal error: Class d contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (c::baz) in **C:\xampp1\htdocs\oop_php\Tugas 3\Lat4_6.php** on line **14**

Error, karena pada object interface, ketika kita mengimplementasikan object tersebut, seluruh method pada interface harus diimplementasikan seluruhnya, Karena class d mengimplementinterface c, maka method-method pada interface c harus diimplementasikan seluruhnnya C

C.



```php
<?php
interface a
{
    public function foo();
}
interface b
{
    public function bar();
}
interface c extends a, b
{
    public function baz();
}
class d implements c
{
    public function foo()
    {
    }
    public function bar()
    {
    }
    public function baz()
    {
    }
}
class e
{
    public function foo()
    {
    }
    public function bar()
    {
    }
}
```

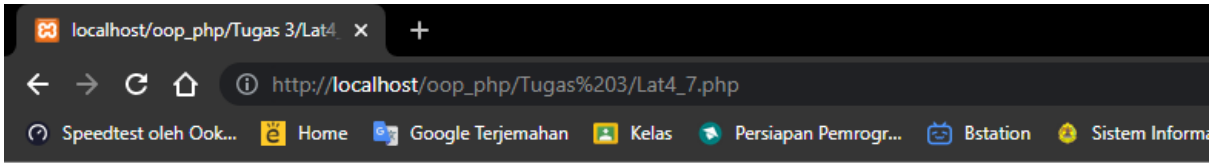
Error, method foo dan bar terdapat pada interface a dan b, pada penerapannya sebuah class baru tidak dapat mengimplementasikan method yang sama pada dua interface

D. Kesimpulan

1) Interface didefinisikan dengan “Interface” keyword, mirip dengan deklarasi class biasa, hanya saja definisi atau detail method tidak dituliskan.
2) Seluruh method yang dideklarasikan pada interface harus memiliki modifier “ public ”.
3) Untuk mengimplementasikan sebuah interface, kita dapat menggunakan “implement” keyword.
4) Seluruh method yang ada pada interface harus diimplementasikan seluruhnya. Sebuahclass bisa mengimplementasikan lebih dari satu interface.
5) Class tidak bisa mengimplementasikan dua interface yang mempunyai nama methodyang sama.
6) Interface bisa diwariskan seperti class menggunakan “extends”.
7) Class yang mengimplementasikan interface harus menggunakan method-method yangada pada interface tersebut dengan nama dan spesifikasi yang sama persis.

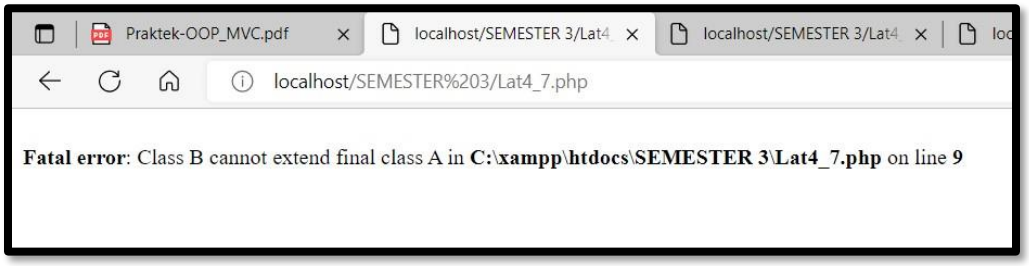
Latihan 7

A. Tampilan program (program tersebut mengalami error)



Fatal error: Class B may not inherit from final class (A) in **C:\xampp1\htdocs\oop_php\Tugas 3\Lat4_7.php** on line **15**

B. Setelah memindahkan kata final dari baris 5 ke baris 2 (program tersebut tetap mengalami error), error pada program tersebut dikarenakan

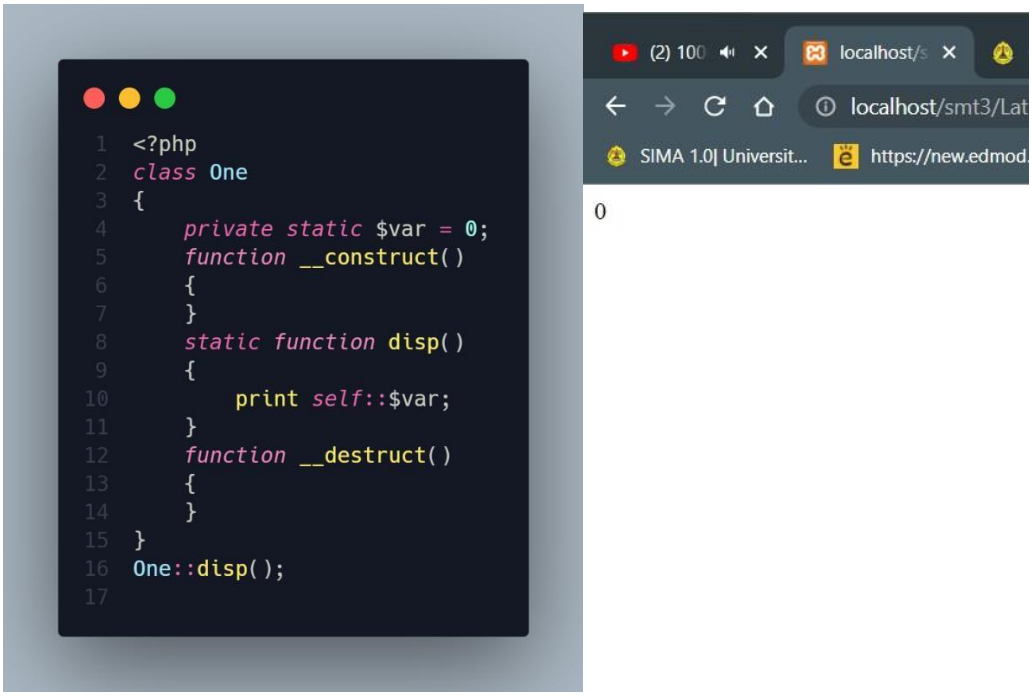


C. Kesimpulan

1. final, akan mencegah proses overriding method pada class anak (sub-class)
2. Apabila metode kita berikan status final, maka metode tersebut tidak akan bisa dioverride, begitu juga pada class, apabila kita berikan status “final” pada deklarasi class maka class tersebut tidak bisa diperpanjang (diwariskan**).**

Latihan 8

A. Tampilan




```php
<?php
class One
{
    private static $var = 0;
    function __construct()
    {
    }
    static function disp()
    {
        print self::$var;
    }
    function __destruct()
    {
    }
}
One::disp();
```

B. Kesimpulan

1. Lat4_8.php mengimplementasikan penggunaan property static dengan modifierprivate.Properti statis dideklarasikan dengan menggunakan katakunci statis sebelum modifier- Sintaks:

   modifier static $nama_property = nilai;

2. Sifat statis dapat diakses tanpa perlu sebuah contoh objek dari kelas, menggunakan nama kelas bersama dengan ::- Sintaks:

   ClassName :: $nama_property method_name();

Perhatikan bahwa properti statis menggunakan tanda dollar ($).Properti statis tidak dapat diakses melalui obyek menggunakan operator **panah "->"**